NoSQL Undo: Recovering NoSQL Databases by Undoing Operations

David Matos, Miguel Correia
INESC-ID, Instituto Superior Técnico, Universidade de Lisboa
Lisboa – Portugal
ICT COST Action IC1405
CONTEXT
NoSQL Databases

- Designed to be distributed
- Abstract definition (unlike SQL databases)
- Key Value vs Document oriented vs graph vs tuple ...
- Relaxed consistency properties (CAP)
- Consistency kept by Global Logs
Most common

mongoDB®

Couchbase

cassandra

ORACLE®

NOSQL DATABASE

Apache HBase

And many more...
NoSQL Common Architecture

NoSQL Database Instance

Router

Router

Shard 1
- Primary
- Secondary
-Secondary
-Secondary

Shard 2
- Primary
- Secondary
- Secondary
- Secondary

Config Servers
- Config
- Config

Support Layer

Intrusion Detector

Global Log

Backup Service

NoSQL Undo
MOTIVATION FOR RECOVERY
Intrusions in NoSQL Databases

• Applications are vulnerable

• SQL Injection and XSS attacks create incorrect records in the database

• NoSQL Databases are not immune to attacks
Removing Intrusions in NoSQL DBs

- Undoing operations is not trivial
- Restoring backups causes data loss
- Correct vs Rolling back
- Current Intrusion Removal methodologies in the literature require software modifications
Rollback Recovery vs Undo

Database before recovery

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Rollback

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After Undo

<table>
<thead>
<tr>
<th></th>
<th>V1</th>
<th>V2</th>
<th>V3</th>
<th>V4</th>
<th>V5</th>
<th>V6</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NOSQL UNDO
Assumptions

• Existing Global Logs to maintain consistency across replicas

• Global Logs can be used to recover the DB

• The DB mechanism for propagating operations can be used for propagating compensation operations

• It is not possible to modify the database software
NoSQL Undo

• NoSQL Undo runs on the client-side

• Uses existing Global Logs to perform recovery

• Does not require modifications to the database software

• An optional Backup Service extends the timespan of recovery

• It can be integrated with an IDS to facilitate the process of identifying malicious operations
NoSQL Undo Architecture

NoSQL Database Instance

- Shard 1
  - Primary
  - Secondary
  - Secondary
  - Secondary

- Shard 2
  - Primary
  - Secondary
  - Secondary
  - Secondary

Config Servers

- Config

Support Layer

- Intrusion Detector
- Global Log Backup Service
Recovery Algorithms

• Full Recovery
  – Recovers by selective re-executing operations

• Focused Recovery
  – More precise algorithm that only “corrects” the incorrect operations by applying Compensating Operations
Full Recovery

• Recovery in three steps (3 R’s)[1]:
  – Rewind
  – Repair
  – Replay

• If there are few incorrect operations to repair it becomes slow

• Requires the database to be unavailable while recovering

Focused Recovery

**In:** faultyOperation

```python
faultyRecord = getRecord(faultyOperation)

if (faultyOperation is INSERT) then
    removeRecord(faultyRecord)
    return

newRecord=[]
operations = getAllOperations(faultyRecord)
operations.remove(faultyOperation)
for each operation in operations
    newRecord.execute(operation) //in memory
end for
```

updateRecord(faultyRecord, newRecord)
PROOF OF CONCEPT - MONGODB
Implementation

• NoSQL Undo was implemented as a Java Application

• It was configured to recover Mongo Databases

• Uses MongoDB Logs (used to keep consistency across replicas)
EXPERIMENTAL EVALUATION
Experimental Evaluation

• Is it advantageous at all to use Full Recovery?

• Is it feasible to recover documents that were modified innumerable times?

• How much is the cost of using an IDS?

• Is it worth it to use the Backup Service?
As the number of incorrect operations increases, Full Recovery becomes a better approach.
Recovering Multi Version Documents

Full Recovery maintains the TTR while Focused Recovery increases exponentially.
The overhead of using an IDS is around between 10% to 30%
Backup Overhead

The overhead of the Backup Service varies from 6% to 30%.
CONCLUSIONS
Conclusions

• NoSQL Undo is capable of removing the effects of undesirable operations in a database

• Both methods are capable of recovering a database

• Focused Recovery is capable of removing an incorrect operation about 1 second

• If more than 60% of the operations in the log are incorrect, then Full Recovery is a better choice
Thank you

David Matos
david.r.matos@tecnico.ulisboa.pt

This work was supported by the European Commission through project H2020-653884 (SafeCloud) and by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 (INESC-ID).